BANCS 接続システム（1）
――プロジェクト成功への鍵：POC 実践報告

Proof of Concept for Mission Critical Accounting System on ES 7000/W2KDCS

山 岸 重 雄

要 約 M 銀行新対外系（BANCS）システムは，世界初の Windows 2000 上での銀行基幹系システムであり，性能面では秒あたり 132 電文の処理，可用性の面では 99.99% の可用率が要求された．ES 7000 と Windows 2000 Datacenter Server は，当該システムへの初めての適用なので，性能，可用性とも未知数であった．そのため，綿密なるテスト計画を立案し，検証作業を行なう事により，早期に開発リスクの把握とリスクが顕在化した場合の適切な対応を行なう必要に迫られた．

本プロジェクトでは開発の各工程毎でのプロトタイピングやシミュレータを開発しての性能測定，米国 UNISYS 社でのベンチマーク等あらゆる角度からの POC（Proof Of Concept）を実施した結果，要求された性能要件，可用性要件が満足できることを検証した．

Abstract The BANCS system for M Bank is the first mission critical accounting system for banking application in the world running on the Windows 2000 system, but that requires for the processing performance of 132 transactions per second and the availability factor of 99.99%. When our project started, there is no measured (or observed) data helpful to estimate the processing performance and availability both of ES 7000 and Windows 2000 Datacenter Server. So, we had to execute POC (Proof of Concept) and we could demonstrate the fact that the proposed system meets our requirements.

This paper describes the case study of POC and shows the performance data of the BANCS system.

1. はじめに

2001 年 4 月，SA 銀行と SU 銀行が合併し，M 銀行が新しく誕生した．合併に伴い，両行でそれぞれ稼働していた様々なコンピュータシステムを一本化することが急務であった．

都銀，および，その他金融機関の現金自動支払機の共同利用による現金支払い業務「BANks Cash Service（略称：BANCS）」を担う対外接続システムも，一本化の対象であった．一本化すると取引量が約 2 倍となり，ピーク時点で 1 秒当たり 132 件のトランザクションを処理する能力が必要となるため，合併前に両行で使用していたシステムでは能力不足が予想された．しかし，両行のシステムともシステム稼働開始から年数を経ており，CPU やメモリの追加などのハードウェアによる増強策は，不可能であった．また，24 時間 365 日稼働などの新サービスの要求もあったため，新たに「新対外（BANCS）システム」（以下，本システム）として，オープン系サーバで再構築することになった．

再構築にあたっては，上述した性能の実現とメインフレーム並みの RAS（Reliability：信頼性，Availability：可用性，Serviceability：保守性）が要求された．その結果，ハードウェアとして Unisys Enterprise Server ES 7000（以下，ES 7000）を採

用し，オペレーティングシステムにはマイクロソフト社の Windows 2000 Datacenter Server（以下，W 2 KDCS）を採用した．

本システムの開発は，2000 年 5 月から 2001 年 10 月までの 1 年半に及び，2001 年 10 月 15 日に予定通り本番を開始した．現在までに大きな問題は発生しておらず，順調に稼働中である．

本システムは，世界で初めて W 2 KDCS 上で銀行基幹業務の勘定系ミッションクリティカルシステムを構築した事例としてエポックメイキングなシステムであり，今後，同様のシステムのリファレンスモデルとなると考えられる．

なお，本稿を含め，以下の 5 編の論文で BANCS システム全体を論じていくが，全体で共通となるシステム概要，システム要件については，本稿の 2 章，3 章で総括的に記述する．

論文名	執筆者
BANCS 接続システム（1） ープロジェクト成功の鍵：POC(Proof of Concept)実践報告	山岸　重雄
BANCS 接続システム（2） ー補完機能開発で実現した HA システム構築	溝上　昌宏
BANCS 接続システム（3） ーホストオンライン処理に要求される信頼性・パフォーマンスの実現	平野　敬幸 安室　秀則
BANCS 接続システム（4） ー大規模トランザクション処理と高可用性を実現した BANCS センタ接続	綾野　昌史
BANCS 接続システム（5） ーマルチプラットフォームサーバ間のデータ通信を実現した MQSeries 適用	赤井　貴

2. 本システムの概要

2.1 本システムの役割

本システムは，自分の口座がある銀行と異なる銀行の CD/ATM（Cash Dispenser/Automatic Teller Machine）を使って，現金の引き出しや残高照会を行なう際に，銀行間で送受信される電文を中継するシステムである（図 1）．

M 銀行を自行，その他の金融機関を他行とすると，他行の CD/ATM で自行の口座を処理する被仕向け処理と，自行の CD/ATM で他行の口座を処理する仕向け処理がある．

本システムと BANCS センタ（他行との中継センタ）との間は，9600 bps の専用回線で「B 改全二重プロトコル」に則り，100 バイトの固定長の電文を送受信する．コード体系は JIS 7 である．

また，勘定系ホストコンピュータとの間は，IBM 社 MQSeries のサーバ機能を備えた FEP 機を介して，平均 1100 バイト（最大 8192 バイト）の可変長の電文を送受信する．コード体系は EBCDIC である．

本システム内では，電文形式や文字コードの変換，入出力電文のロギング，指定口座の名義人名の検索などホストコンピュータで処理される業務の一部代行を行なう．

2.2 システム構成

本システムでは，地震などの災害で運用センタが使用できなくなった場合(被災時)

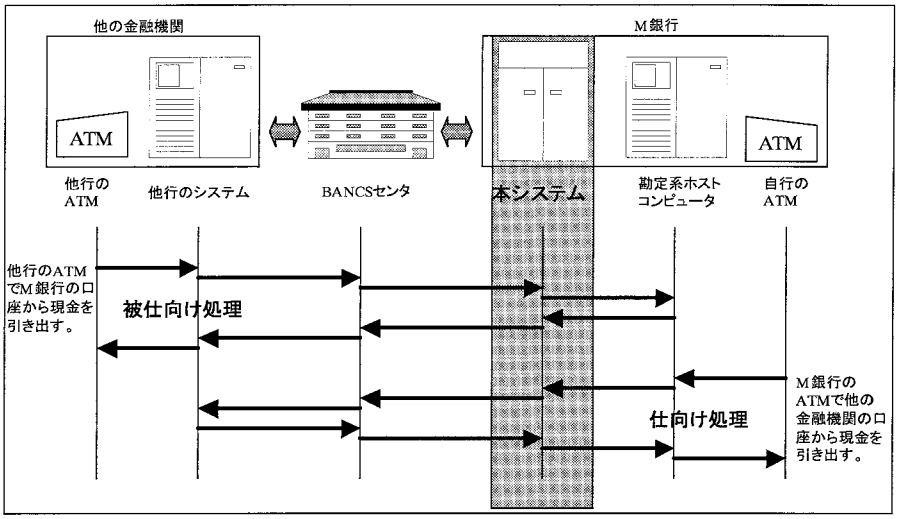

図 1　本システムの位置づけ

にも，業務の全面停止にならないように関東地区と関西地区に運用センタを配置する．また，各センタ内では，2 台の ES 7000 で Active/Active のクラスタを構成し，通常時は負荷分散を図り，障害発生時は 1 台の ES 7000 での運用を可能にしている（図 2）．

BANCS センタとの接続回線は，各センタで直通回線 4 本，中継センタ経由 4 本の計 8 本で接続されている．被災時は通常時の 8 本に加えて，被災したセンタに接続されていた中継センタ経由の 4 本が切り替えられ，合計 12 本での接続となる．

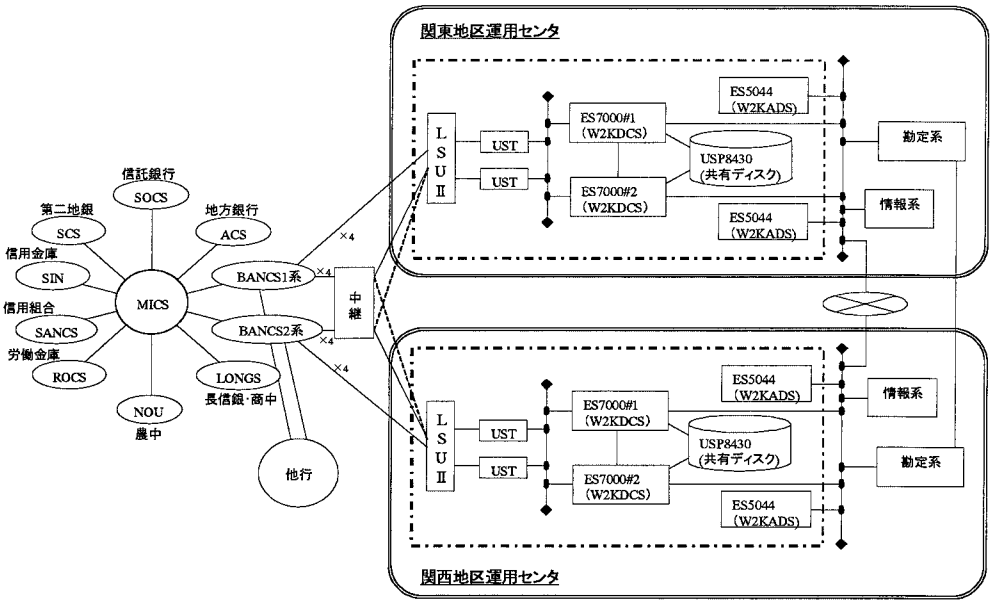

図 2　システムの全体構成

2.2.1 適用ハードウェア

本システムで使用するハードウェアのうち，主だったものを表 1 に示す．

表 1　本システムで使用する主要ハードウェア

役　割	使用機器	センタあたりの台数	詳　細
BANCS サーバ	ES7080SD	2	CPU：Pentium III Xeon 700MHz×8 メモリ：4GB
共有ディスク	UPS8430	1	EMC 製 Symmetrix，36GB×16
回線切替装置	LSU-II plus	1	UST 障害発生時に，正常な UST に切り替える．
通信制御装置	UST SC-8270	2	セイコープレシジョン製 B 改全二重プロトコルで送受信している通信回線上の電文を TCP/IP のパケットに変換する．
ドメインコントローラ	ES5044	2	本システムのドメインを管理する．

2.2.2 適用ソフトウェア

BANCS サーバで使用するソフトウェアを表 2 で示す．

表 2　BANCS サーバ上のソフトウェア

役　割	ソフトウェア名称	詳　細
OS	W2KDCS	システム実行時の基本機能を提供する． クラスタ構成は，W2KDCS の標準機能であるクラスタサービスを使って構築する．
DBMS	SQL Server 2000	共有ディスク上に 2 つのデータベースを構築し，通常運用時は 2 台の BANCS サーバ上の SQL Server から各データベースを使用する．フェイルオーバ運用時は，1 台の BANCS サーバ上で 2 つの SQL Server が稼動し，各データベースを使用する． 被災運用時は，通常運用時と同じデータベースを使用する．
BANCS センタ接続 I/F	WinSAM	UST との通信を行なう．
	BANCS リスナ	WinSAM 経由で受信した BANCS センタからの電文を，BANCS アプリケーションに渡す．回線毎に 1 プロセスを割り当てる．
	BANCS センダ	BANCS アプリケーションから渡された電文を，WinSAM 経由で BANCS センタに送信する．回線毎に 1 プロセスを割り当てる．
勘定系 FEP 接続 I/F	MQ FEP 接続 I/F	IBM MQSeries のサーバである勘定系 FEP に MQ クライアントとして接続し，電文を送受信する．勘定系 FEP のヘルスチェックも行なう．勘定系 FEP 毎に 1 プロセスを割り当てる．
BANCS アプリケーション	一般電文処理	BANCS センタから受信した一般電文を処理する．
	制御電文処理	BANCS センタから受信した制御電文を処理する．
	回線障害電文処理	BANCS センタに送信する際に回線障害が発生した電文を処理する．
	勘定系電文処理	勘定系ホストから受信した電文を処理する．
	タイマー処理	タイムアウトした電文を処理する．
	コマンド処理	コマンドを処理する．
	モニタ処理	各プロセスの実行状況を監視する．

上記のソフトウェアが稼働している状態を図 3 に示す．

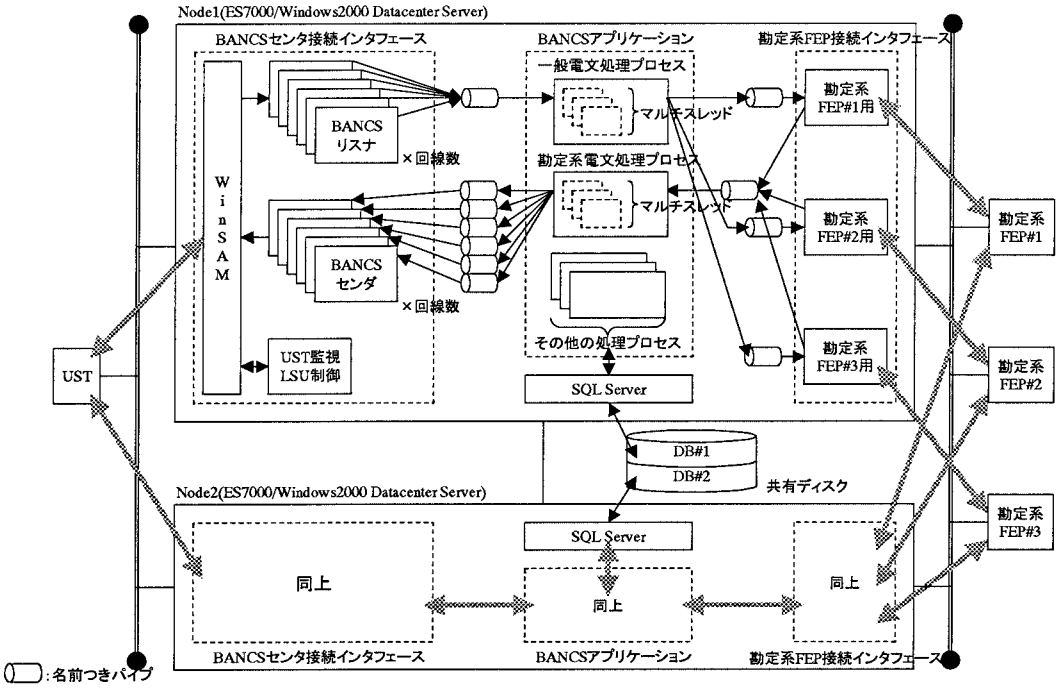

図 3　サーバ内のソフトウェア構成

BANCS センタから送られてきた電文の流れは以下のとおりである.

① BANCS センタから送られてきた電文は，UST がいったん受信し，UST が電文内の経路番号により，2 台の BANCS サーバのどちらに電文を送信するかを振り分ける.

② BANCS サーバ内では，WinSAM の受信関数を呼んで待機状態にいる BANCS リスナが UST から送られてきた電文を受信し，一般電文処理または制御電文処理用の BANCS アプリケーションに送信する.

③ BANCS アプリケーションでは，電文の内容によりデータベースをアクセスし，電文形式を変換して，MQ FEP 接続 I/F に電文を送信する.

④ MQ　FEP 接続 I/F は，IBM　MQSeries のサーバである勘定系 FEP 内の Queue に MQSeries のクライアント API を使用して電文を PUT する.

勘定系ホストコンピュータから送られてきた電文の流れは以下のとおりである.

① 勘定系ホストコンピュータから送られてきた電文は，勘定系 FEP 内の Queue にいったん蓄えられる.

② BANCS サーバ内では，MQ FEP 接続 I/F が MQSeries のクライアント API を使用して勘定系 FEP 内の Queue から電文を GET し，勘定系電文処理用の BANCS アプリケーションに送信する.

③ BANCS アプリケーションでは，電文の内容によりデータベースをアクセスし，電文形式を変換して，BANCS センダに電文を送信する.

④ BANCS センダは，WinSAM の送信関数を使用して，UST 経由で BANCS センタに電文を送信する.

各プロセス間は，名前付きパイプを使用して電文の受け渡しを行なう．

3. 本システムで求められるシステム要件

本システムは，銀行基幹業務の勘定系ミッションクリティカル・システムであり，性能，信頼性，可用性，保守性が厳しく要求された．以下，各要件について記述する．

3.1 性 能 要 件

要件定義で要求された性能要件を表 3 に示す．なお，図 1 で示したように，1 業務処理では上りの要求電文と下りの回答電文の 2 電文を処理するが，1 電文につき 1 トランザクションが発生する．

銀行提示の必要処理業務件数（ピーク時）より，通常時は，4 台のサーバ（本システム全体）で 132 trx/sec を処理する必要がある．したがって，1 台あたりは 33 trx/sec となる．

被災運用時は 1 センタでの運用になるが，この場合，他センタ分の業務は縮対運用のため通常処理量の半分になるので，1 台あたりは 33 trx/sec×1.5 の 50 trx/sec となる．

また，センタ内で 1 台のサーバで障害が発生するとフェイルオーバが発生し，残りの 1 台での運用になるが，この場合は 1 台で 2 台分の処理をする為，33 trx/sec×2 の 66 trx/sec となる．被災運用状態でフェイルオーバが発生した場合は，50 trx/sec ×2 の 100 trx/sec となる．

最悪のケースを想定すると，1 台のサーバで 100 trx/sec を満足する必要がある．

表 3　性能要件

項目	内　　容	目標値
滞留時間	1 業務処理あたりの本システム内滞留時間	平均 1 秒以内
処理量	本システム全体	66 業務処理/秒 (132 trx/sec)
	サーバ 1 台当たり　■通常時	16.5 業務処理/秒 (33 trx/sec)
	同　　■被災運用時	25 業務処理/秒 (50 trx/sec)
	同　　■フェイルオーバ時	33 業務処理/秒 (66 trx/sec)
	同　　■被災＆フェイルオーバ時	50 業務処理/秒 (100 trx/sec)
リソース使用量	被災時	CPU：40%以内 メモリ：70%以内
	フェイルオーバ時	CPU：53%以内 メモリ：92%以内

3.2 信頼性要件

障害（ハードウェアの故障，ソフトウェアの不具合など）が発生しないこと．

3.3 可用性要件

不幸にして障害が発生してしまった場合にも，システム停止時間を最小限にすること．

要件定義で要求された可用性要件を以下にあげる．

1） 24 時間 365 日運転

障害の早期検出と自動復旧によって，システム停止時間を最小限にし，可用性を高めること．システムの構成変更，運用変更に柔軟に対応できること．

2） 冗長構成

ハードウェア，ソフトウェアの各部位において可能な限り冗長構成をとり，システムの可用性を高めること．

3） 即時バックアップ

システムはクラスタリング構成をとり，片系のサービスが停止しても，他系が自動的かつ速やかに処理を引き継ぐこと．

4） 被災対応

地震，火災などの災害やシステム全体が停止するような重大障害が発生した場合に備えて，地理的に離れた 2 つの拠点（センタ）に同一システムを設置し，被災時に別のセンタにあるシステムが処理を引き継ぐことができること．

可用性要件の数値目標を表 4 に示す．

表 4　可用性要件

項目	目標値
センタあたりの可用率	99.99%（24 時間 365 日で 50 分以内のシステム停止）
サーバあたりの可用率	99.94%（24 時間 365 日で 315 分以内のシステム停止）

また，障害発生した時点から 3 分以内にフェイルオーバなどでの自動復旧処理が完了し，業務を完全に復旧させることが要求された．

3.4　保守性要件

障害が発生した場合，原因の追求が可能であること．

■各論文の位置づけ

1 章で述べたように，5 編の論文を通して BANCS システムを論じていくが，各論文は以下の観点から論述されている．

1）「プロジェクト成功の鍵：POC 実践報告」

ES 7000 と W 2 KDCS の組み合わせが，基本的な性能要件，信頼性要件，可用性要件を満足することができる，ということをどのようにして実証したか．

2）「補完機能開発で実現した HA システム構築」

ES 7000 と W 2 KDCS の組み合わせが，如何にして更なる高可用性要件を実現したか．

3）「ホストオンライン処理に要求される信頼性・パフォーマンスの実現」

業務ロジック部分の性能要件，信頼性要件，可用性要件，保守性要件を実現するため，ミドルウェアを開発した．その機能と仕組みは如何なるものか．

4）「大規模トランザクション処理と高可用性を実現した BANCS センタ接続」

BANCS センタとの接続インタフェース部において，如何にして性能要件，可用性要件を実現したか．

5）「マルチプラットフォームサーバ間のデータ通信を実現した MQSeries 適用」

勘定系ホストコンピュータの FEP である MQ サーバ機との接続インタフェース部において，如何にして性能要件，可用性要件を実現したか．

4. POC の実施

POC とは Proof Of Concept の略で，ハードウェア構成，ネットワーク構成，ソフトウェア構成，プロセス/スレッド構成などのシステムアーキテクチャに問題がないかどうかを検証する作業である．プロジェクトの早い段階から POC を実施することで，リスクを正確に把握することができる．

本プロジェクトでは，要件定義工程から POC を実施し，「システム要件を満足するためには何をすべきか」ということを分析し，成果物にフィードバックすることで，最終的にはシステム要件を達成することができた．「システム開発の各工程で，どのような POC を実施したか」を以下に説明する．

4.1 要件定義工程

目的：本システムで使用する基盤ソフトウェアの基本性能，問題点を確認する．

時期：2000 年 5 月～6 月

内容：本システムは基盤ソフトウェアとして，OS に W 2 KDCS,DBMS に SQL Server 2000,BANCS センタとの接続 I/F として WinSAM，勘定系 FEP との接続 I/F として IBM　MQSeries を使用する．これらのソフトウェアの基本性能，問題点を確認した．ただし，この時期は ES 7000 および W 2 KDCS が製品版リリース前であり，使用環境がなかったため，ES 5000 と Windows 2000 Advanced Server で検証を行った．また，WinSAM も本番で使用する予定の V 5 版がリリース前であったため，V 4 M 2 版を使用した．UST も個別改造の設計段階であったため, UST を使わず WinSAM 対向による LAN 接続で検証した．

■性能要件の POC

① 本システムの基盤ソフトウェアについて，各製品毎に基本性能を測定した．MQSeries については，サーバ接続とクライアント接続の比較を行った．

■信頼性要件の POC

① SQL Server 2000,WinSAM,IBM MQSeries が Windows 2000 上で問題なく動作することを検証した．

② SQL Server 2000 のデッドロックが，どういう場合に発生するかを検証した．

■可用性要件の POC

① Windows 2000 のクラスタサービスで本システムの可用性を支えることが可能かどうか，Active/Active のクラスタ構成が実現可能か，を検証した．

② SQL　Server 2000,WinSAM,IBM　MQSeries がクラスタサービスに対応しているかどうかを検証した．

③ ネットワークの 2 重化を実現する NIC（Network Interface Card）ドライバの AFT（Adapter Fault Tolerance）機能が Windows 2000 上で問題なく動作するか，クラスタサービスとの相性が良いかどうかを検証した．

4.2 論理設計工程

目的：1 電文を処理する時間や秒あたりに処理できる電文数などのおおよその性能を掴む.

時期：2000 年 7 月～8 月

内容：本番と同じ長さの電文を受信し，本番と同程度の DB アクセスを行い，電文を送信するプロトタイプ 1 を SQL Server 2000,WinSAM,IBM MQSeries を組み合わせて作成し，性能，問題点を確認した．BANCS センタ側の動きを担う UST シミュレータ，勘定系 FEP 側の動きを担う MQFEP シミュレータも併せて開発した．負荷テストツール LoadRunner を使用して，電文数の制御や応答時間，リソース使用状況を確認した.

しかし，まだ ES 7000/W 2 KDCS が製品版リリース前であったため，要件定義工程と同様の環境で検証を行った.

4.3 物理設計工程

目的：ES 7000 と W 2 KDCS での性能検証と可用性検証で，問題の有無と課題を明確にする.

本番と同等のソフトウェア構成，プロセス構成，スレッド構成での問題点の有無と課題を明確にする.

時期：2000 年 9 月～10 月

内容：システム開発及び基盤開発，検証用 ES 7000 2 台が日本ユニシス（以下，当社）の本社データセンタへ搬入（9 月）され，本格的なテスト環境構築およびテストを実施した．並行して，客先納入向け ES 7000 を使用した米国（Mission Viejo）でのベンチマークを実施した.

米国および日本でのテストは，その主たる目的を以下のように分け，効率化を狙った.

・国内：基盤系結合テスト，米国テスト結果を受けての性能詳細評価.

・米国：性能測定およびチューニング，ハードウェア系障害テスト，ロングランテスト，客先納入機器のステージング.

本番と同等のソフトウェア構成，プロセス構成，スレッド構成（2.2.2 項参照）であるプロトタイプ 2 での検証を実施した．また，WinSAM の V 5 版がリリースされたので，これを使用した検証を実施した.

■性能要件の POC

① ES 7000/W 2 KDCS での性能検証を実施した．ES 5000 と比較して CPU 使用率が高いことが判明した.

② 本番同等のソフトウェア構成であるプロトタイプ 2 を使用し，性能検証を実施した.

③ WinSAM V 5 版での性能検証を実施した．V 4 M 2 版に較べて性能が悪いことが判明した.

■信頼性要件の POC

① プロトタイプ 2 及びシミュレータにより 2 週間の高負荷（100 trx/sec）連続稼働を実施し，安定性を検証した.

■可用性要件の POC

① ハードウェア障害が発生した際に，冗長構成で実現しているバックアップの仕組みが正しく機能することを検証した．また，その際の切替時間を計測した．

② 負荷をかけながら障害を発生させて，フェイルオーバ後も処理が継続して実行できることを検証した．

4.4 プログラム開発工程

目的：性能のボトルネックの確認．UST の検証．

時期：2000 年 11 月～12 月

内容：サーバ内滞留時間をより詳細に把握し，性能ネックはどこかを調査するため，BANCS 電文にタイムスタンプを埋め込む機能を付けたプロトタイプ 2 を使用して，プロセス毎の処理時間を計測した．

また，BANCS 向け個別改造版の UST も当社の本社データセンタへ搬入されたので，UST を使用した回線経由での性能検証を実施した．

■性能要件の POC

① タイムスタンプ埋め込み機能付のプロトタイプ 2 を使用し，サーバ内のどこで処理時間が大きいかを確認した．

② UST を使用した回線経由での性能を検証した．

■可用性要件の POC

① フェイルオーバの際に，UST が電文を送信する相手の IP アドレスが切り替わることを確認した．

4.5 統合，システムテスト工程

目的：本番環境での性能検証，可用性検証

時期：2001 年 1 月～13 年 6 月

内容：

■性能要件の POC

① 本番環境と同等のハードウェア，ソフトウェア，ネットワーク構成で性能要件を満たすことができるかどうかを検証した．

■信頼性要件の POC

① 本番と同様の運行スケジュールで 2 週間の連続稼働を実施し，安定性を確認した．

■可用性要件の POC

① ハードウェア，ソフトウェア，ネットワークを問わず，システムの構成要素で障害が発生した際に，冗長構成で実現しているバックアップの仕組みが正しく機能することを検証した．また，その際の切り替え時間を計測し，可用性要件を満足することを確認した．

② システム破壊が発生した場合，バックアップしたシステムをリストアして，正常に復旧できるか．また，復旧までに時間はどれくらいかかるかを確認した．

5. 検 証 結 果

5.1 信頼性要件と可用性要件の POC

本システムでの基盤となる，OS，データベースソフトウェア，WinSAM,IBM MQ クライアントについて，基本機能の確認を実施したが，特に重要視した機能についてのテスト結果を以下に示す．

なお，可用性要件の検証結果については，本誌掲載の論文「BANCS 接続システム(2)ー補完機能開発で実現した HA システム構築（溝上昌宏著)」に詳しく記述されているので参照されたい．

1）Windows 2000

NIC の 2 重化	Intelドライバの AFT(Adapter Fault Tolerance)機能で対応可能．NIC 切替時間は約 1 秒．AFT による NIC 切替は，Cluster Service では障害とみなさない．2 枚の NIC 障害で，障害とみなす．
Active/Active でのクラスタ構成	特に問題なし．

2）SQL Server 2000

デッドロック	**5.2 節で詳しく述べる．**
クラスタ上での動作	特に問題なし．

3）WinSAM

W2K 上での動作	特に問題なし．
クラスタ上での動作	複数インスタンスは不可なので，フェイルオーバの対象にはできない． WinSAM 起動時に IP アドレスが使用できる状態になっている必要があるので，途中からフェイルオーバしてくる仮想 IP アドレスは使用できない． ということで，WinSAM はフェイルオーバしないことにして，フェイルオーバ時に UST からの電文送信先を変更するために，「UST 振分 IP アドレス設定プログラム」を開発することにした．

4）IBM MQ クライアント

W2K 上での動作	特に問題なし．
クラスタ上での動作	特に問題なし．

5.2 SQL Server のデッドロック

デッドロックは，二つのトランザクション間で相互にロックし合うことで発生する現象である．いわゆる「たすきがけ」と呼ばれている状況が起こると発生するが，SQL Server の場合はそれ以外にも，以下の二つのケースでデッドロックが発生することが確認された．

5.2.1 変換デッドロック

ロックモードが「共有」から「排他」に変換される際に発生するデッドロックで，SELECT で参照した行を，UPDATE で更新する際に発生する．

例えば，以下のトランザクションを実行すると，②のタイミングで table 1 の index 1 が 123 の行（「行 123」とする）に対し共有ロックがかかる．

① BEGIN TRANSACTION

② SELECT column 1 FROM table 1 WHERE index 1＝123

③　UPDATE table 1 SET column 1＝111 WHERE index 1＝123
④　COMMIT TRANSACTION

そして，③のタイミングで共有ロックから排他ロックに変換される．

このトランザクションを同時に二つ（A と B）実行すると，

A①：トランザクション A が開始される．
B①：トランザクション B が開始される．
A②：トランザクション A が，行 123 に共有ロックをかける．
B②：トランザクション B が，行 123 に共有ロックをかける．
A③：トランザクション A が，行 123 に排他ロックをかけようとするが，トランザクション B が共有ロックをかけているので，待ちになる．
B③：トランザクション B が，行 123 に排他ロックをかけようとし，デッドロックが発生する．

となる．

この現象は，②で SELECT する際に

②'SELECT column 1 FROM table 1 WITH（UPDLOCK）WHERE index 1＝123

と明示的に排他ロックをかけるようにすることで回避できる．

5.2.2　ロックエスカレーションによるデッドロック

SQL　Server のロックの粒度には，行，ページ（8 KB），テーブルがあり，ロックをかけたときにどの粒度になるかは自動的に決まる．このことが原因でデッドロックが発生することがある．

例えば，トランザクション A が

①　BEGIN TRANSACTION
②　SELECT column 1 FROM table 1 WITH（UPDLOCK ROWLOCK）WHERE index 1＝123
③　UPDATE table 1 SET column 1＝111 WHERE index 1＝123
④　COMMIT TRANSACTION

で，トランザクション B が

①　BEGIN TRANSACTION
②　SELECT column 1 FROM table 1 WITH（UPDLOCK ROWLOCK）WHERE index 2＝120
以後，省略

としたとき，index 1 をキーとする索引は作成してあるが，index 2 をキーとする索引が作成してない場合，二つのトランザクション A,B を同時に実行すると，

A①：トランザクション A が開始される．
B①：トランザクション B が開始される．
A②：トランザクション A が，行 123 に排他ロックをかける．
B②：トランザクション B が，index 2＝120 に該当する行に排他ロックをかけようとするが，index 2 に索引がないので全件検索する．行 120 は該当したので排他ロックをかけるが，行 123 は既にトランザクション A が排他ロックをかけているので条件に該当するかどうか調べる動作が待ちになる．

A③：トランザクション A が，行 123 を更新しようとするが，UPDATE 文に明示的に行ロックの指定がないので，ページに対して排他ロックをかけようと試みる．このとき，行 123 と行 120 が同じページであると，行 120 はトランザクション B が排他ロックをかけており， デッドロックが発生する．

となる．

この現象は，index 2 に索引を作成するとともに，③で UPDATE する際に

③ 'UPDATE table 1 WITH（ROWLOCK）SET column 1＝111 WHERE index 1＝123

と明示的に行ロックをかけるようにすることである程度回避できる．しかし，明示的にロックの粒度を指定しても，SQL Server が勝手にロックの粒度をエスカレーションさせてしまうことがあり，常に行ロックがかかるという保障がない．「デッドロックは発生するものだ」という前提で，発生した際はトランザクションをリトライするなどの考慮が必要である．

5.3 性能要件の POC

5.3.1 性能要件の達成

以下の環境で性能検証を実施し，性能要件を満たすことを確認した．

1） 使用ハードウェア

性能検証で使用したハードウェアを表 5 に示す．また，システム構成図を図 4 に示す．

表 5 性能検証で使用したハードウェア

BANCS サーバ 1 号機	ES7080SD(P Ⅲ Xeon 700MHz×8, MEM 4GB) 注 1
BANCS サーバ 2 号機	ES7080SD(P Ⅲ Xeon 700MHz×8, MEM 4GB) 注 1
共有ディスク	EMC8430
UST	SC-8270
LSU	LSU-Ⅱ plus
UST シミュレータ機	ES5000(P Ⅲ Xeon 550MHz×4, MEM 2GB)
MQFEP シミュレータ機	ES5000(P Ⅲ Xeon 550MHz×4, MEM 2GB)
LoadRunner Controller	ES2000(P Ⅲ 550MHz×2, MEM 2GB)

注 1 : ES 7080 SD はこの時点でのハードウェアの制限事項により 650 MHz で稼働

2） 使用ソフトウェア

性能検証で使用したソフトウェアを表 6 に示す．

3） CPU Affinity の設定

ES 7000 は，四つの CPU と Third Level Cache で SUB-POD を構成している．今回使用した ES 7080 SD の 8 CPU 構成の場合，二つの SUB-POD から構成されている．各 CPU がメモリアクセスする際に，まず，プロセッサ内にあるキャッシュを参照し，ヒットしなければ，SUB-POD 内の Third Level Cache を参照する．Third Level Cache でもヒットしなかった場合，メインメモリにアクセスしにいくが，このとき，他の SUB-POD 上の Third Level Cache 上に存在していると，メインメモリにアクセスするときよりも時間がかかる．メモリ間データ

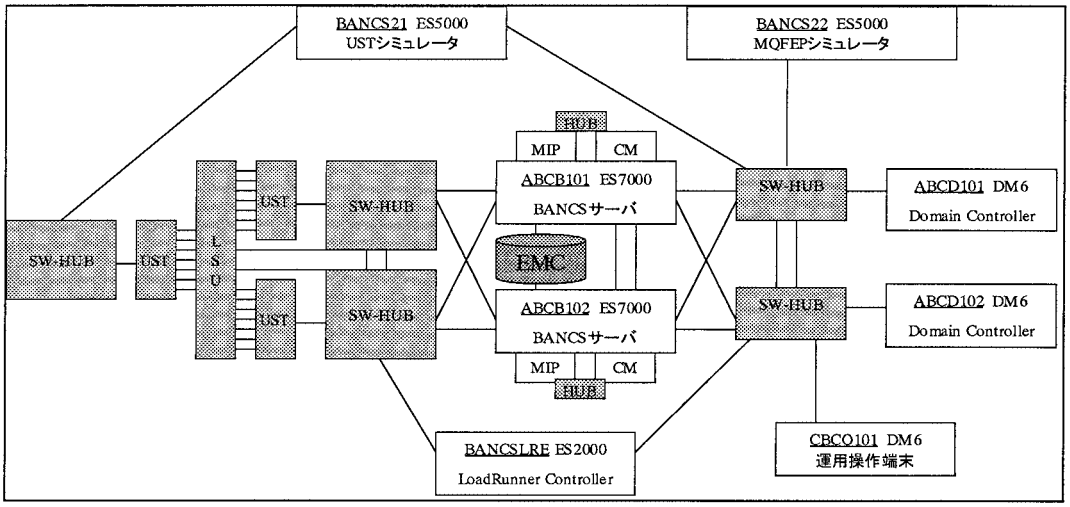

図 4　性能検証のシステム構成

表 6　性能検証で使用したソフトウェア

Plateau	8.0 IC1
W2KDCS	5.0.2195 Service Pack 1 ビルド 2195
SQL Server	8.00.194
WinSAM	V5M1FE
IBM MQ Series	5.1CSD06
BANCS Application	RLS2.1-02
擬似 BANCS Application	RLS2.1-02(タイムスタンプ採取機能付き)
UST シミュレータ	電文発行:被仕向け支払い要求, 仕向け支払い回答
MQFEP シミュレータ	電文発行:仕向け支払い要求, 被仕向け支払い回答

転送を頻繁に行なうプロセス同士は Third Level Cache を共用するように CPU Affinity（各スレッドは，そのスレッドが実行することを許されたプロセッサを指定する Affinity マスクを持つ）を設定することにより CPU を効率よく使用でき，性能が向上することが情報としてあったため，米国でのベンチマークで各種のチューニングを実施し，その効果について確認を行なった．

米国での測定結果をもとに，日本で更にチューニングを実施し，最終的に．WinSAM と BANCS センダ，リスナのように互いに親和性の高いプロセスを，同じ SUB-POD 上で実行するよう，表 7 のように設定した．

表 7　設定した CPU Affinity

使用する CPU#	プロセス
0,1,2,3	SQL Server, MQFEP I/F, BANCS AP1
4,5,6,7	WinSAM, BANCS リスナ, センダ, BANCS AP2

本システムモデルでは，CPU Affinity の設定により，20% の CPU 使用率の軽減につながった．詳細な性能比較については，5.3.3 項で詳述する．

4） 計測結果

計測結果を表 8 に示す．表に示すように，性能要件を満たすことが確認できた．

表 8 性能検証の計測結果

	通常時		被災時		フェイルオーバ時	被災＆フェイルオーバ時
1号機	1#1		1#1	2#1		
2号機	1#2		1#2	2#2	1#1 1#2	1#1 2#1 1#2 2#2
■性能要件						
Trx数(trx/sec)	33		50		66	100
CPU使用率(%)	−		40.0		53.0	−
メモリ使用率(%)	−		70.0		92.0	−
滞留時間(ms)	1000		1000		1000	−
■測定値(本番APで測定)						
発生Trx数(trx/sec)	35.3	36.2	50.9	52.7	68.9	100.8
CPU使用率(%)	**20.9**	**21.4**	**35.6**	**34.4**	**41.8**	**84.3**
メモリ使用率(%)	21.0	29.6	21.2	29.8	33.3	41.3
応答時間(ms)						
被仕向け	915		956		926	1238
仕向け	970		1039		956	1387
滞留時間(ms) = 応答時間 − シミュレータでのsleep時間(500ms) − 回線上の伝送時間(200ms)						
被仕向け	**215**		**256**		**226**	**538**
仕向け	**270**		**339**		**256**	**687**
プロセス毎のCPU使用率						
SQL Server	16.1	17.6	30.4	30.2	16.9	38.6
WinSAM	3.8	3.5	8.3	7.7	9.1	21.5
BANCS AP1	52.8	53.3	31.8	59.6	58.3	75.3
BANCS AP2	45.6	47.6	30.3	54.9	54.2	65.8
BANCSリスナ	0.5	0.8	0.8	1.0	0.8	0.6
BANCSセンダ	0.9	0.6	0.5	0.8	0.7	0.8
MQFEP I/F	0.8	0.7	0.6	0.9	0.8	2.2
プロセッサ毎のCPU使用率						
Processor 0	22.2	16.2	36.5	26.7	43.9	89.2
Processor 1	18.7	19.0	37.0	34.5	37.9	90.5
Processor 2	24.5	28.5	35.8	42.7	46.1	89.0
Processor 3	23.4	27.2	39.0	38.5	45.9	89.0
Processor 4	15.2	13.7	28.3	28.8	36.8	78.3
Processor 5	17.3	18.5	31.3	31.0	36.0	76.8
Processor 6	21.4	20.7	35.9	35.7	42.0	82.0
Processor 7	24.1	27.4	41.4	37.1	45.6	79.5
Context Switch/sec	6983	6263	9249	8327	9837	13425

5.3.2 最大可能処理量

この構成での最大可能処理量（ES 7080 SD の性能限界値）を求めるため，シミュレータから送信する電文数を増加させて限界点を計測した．この計測は，各プロセス内での経過時間を測定するためタイムスタンプ採取機能を付加した擬似 AP を使用した．結果が表 9，図 5 であり，110 trx/sec が限界であることがわかった．

この限界の主要因は回線である．

回線数が 8 の場合は 76.1 trx/sec が最大であったが，回線数を 12 に増やしたところ，110.6 trx/sec まで増加した．1 回線あたり約 9 trx/sec の処理が限界である．

1 回線の最大スループットを調べた結果, 表 10 に示すように 9.15 trx/sec であった．

また，負荷が高くなった際に，どのプロセスで時間がかかっているかを調べるためにタイムスタンプ埋め込み機能つきの擬似 AP を使用して，図 6 に示した区間の処理

表 9　トランザクションの逓増

trx/sec	20	40	60	76	80	100	110
回線数	8	8	8	8	12	12	12
計測結果							
業務件数／秒	9.6	19.3	29.0	36.9	40.1	49.6	53.9
被仕向け	6.1	11.4	17.7	22.8	23.2	26.8	28.5
仕向け	3.6	7.9	11.3	14.1	16.9	22.8	25.4
SQL Serverのtrx/sec	20.7	40.7	60.4	76.1	81.1	100.7	110.6
応答時間(ms)							
被仕向け	811	828	859	891	910	955	1137
仕向け	822	849	954	1394	1288	1260	1455
滞留時間(ms) = 応答時間 − シミュレータでのsleep時間(500ms) − 回線上の伝送時間(200ms)							
被仕向け	111	128	159	191	210	255	437
仕向け	122	149	254	694	588	560	755
CPU使用率(%)							
BANCSサーバ	14.4	26.4	39.7	50.0	56.9	72.2	80.4
User	8.0	15.1	22.8	30.8	34.4	43.0	47.3
SQL Server	7.5	22.5	29.8	41.0	40.5	56.1	65.5
WinSAM	2.7	6.2	10.4	12.7	14.6	20.3	23.2
BANCS AP1	30.9	60.3	97.9	127.0	133.7	141.4	139.4
BANCS AP2	32.7	58.4	87.5	122.3	133.2	135.4	135.1
BANCSリスナ	0.5	0.6	0.7	1.2	0.8	0.6	0.7
BANCSセンダ	0.5	0.5	0.8	0.9	0.9	0.9	1.0
MQFEP I/F	0.6	1.4	4.7	2.0	1.9	1.9	1.7
Context Switch/sec	4315	6798	9141	10987	11626	13515	14235

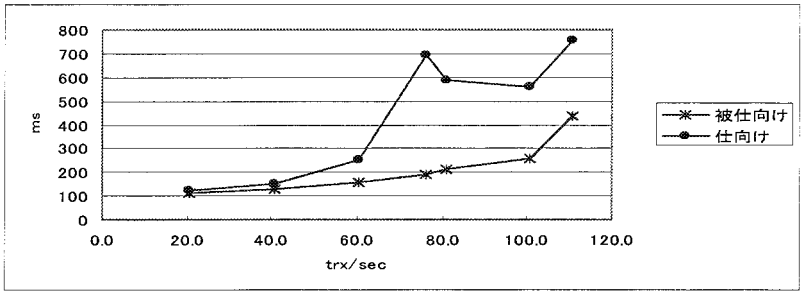

図 5　trx/sec と滞留時間

表 10　1 回線での限界値

多重度	応答時間	trx/sec
1	961	2.1
2	950	4.2
3	977	6.13
4	1088	7.35
5	1093	9.15
6	1317	9.15
7	1531	9.13

時間を計測した．計測結果を表 11 に示す．

表 11 および図 7 に示す各プロセスの処理時間の計測結果から，処理量が 100 trx/sec から 110 trx/sec に増えた際に，電文が BANCS AP に入ってから出るまでの時間が急激に増加していることがわかる（被仕向けの④と⑩，仕向けの④と⑩）．

擬似 AP の④と⑩での内部処理は，大半が SQL Server のアクセスである．したがって，この時間は AP が SQL Server に対して SQL 文を投げて結果が返ってくるまでの時間であるとみなせる．

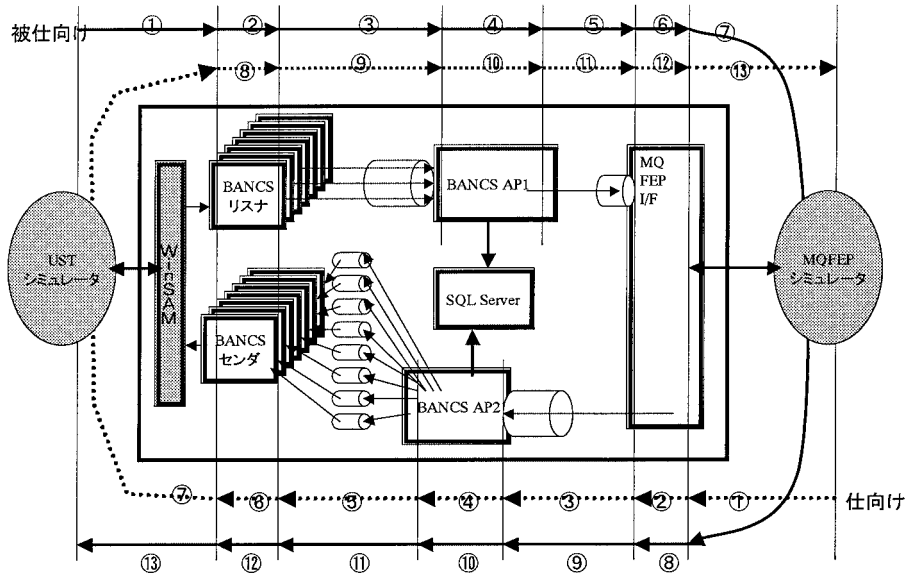

図 6　処理時間を計測した区間

表 11　各プロセスの処理時間

被仕向け(In-bound)	20	40	60	76	80	100	110	
① シミュレータ→BANCSリスナIn	127.5	133.4	144.9	155.3	159.0	158.3	177.1	
② BANCSリスナIn→Out	0.0	0.0	0.0	0.0	0.0	0.0	0.1	
③ BANCSリスナOut→BANCS AP1 In	0.2	0.2	0.2	0.4	0.7	1.3	4.0	
④ BANCS AP1 In→Out	42.5	47.4	55.0	60.5	67.8	85.8	**150.8**	*
⑤ BANCS AP1 Out→MQ I/F In	0.2	0.2	0.3	0.3	0.5	1.1	3.2	
⑥ MQ I/F In→Out	0.1	0.1	0.1	0.2	0.3	0.4	0.7	
⑦ MQ I/F Out→シミュレータ→MQ I/F In	497.2	497.8	497.8	498.4	498.9	499.3	501.0	
⑧ MQ I/F In→MQ I/F Out	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
⑨ MQ I/F Out→BANCS AP2 In	0.2	0.2	0.3	0.5	0.8	1.9	4.8	
⑩ BANCS AP2 In→Out	40.0	43.7	51.7	57.8	64.1	84.4	**139.1**	*
⑪ BANCS AP2 Out→BANCSセンダIn	0.1	0.1	0.3	0.3	0.6	1.4	3.0	
⑫ BANCSセンダIn→BANCSセンダOut	0.1	0.3	1.0	6.5	7.1	9.3	35.6	
⑬ BANCSセンダOut→シミュレータ	102.6	104.5	107.5	111.0	110.3	111.5	117.8	

仕向け(Out-bound)	20	40	60	76	80	100	110	
① MQFEP Sim→MQ Listener In	0.1	0.1	0.1	0.2	0.3	0.4	0.7	
② MQ Listener In→MQ Listener Out	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
③ MQ Listener Out→BANCS AP② In	0.1	0.2	0.4	0.5	0.9	1.9	4.6	
④ BANCS AP② In→Out	49.0	52.2	59.1	64.3	71.1	95.4	**149.9**	*
⑤ BANCS AP② Out→BANCSセンダIn	0.1	0.1	0.2	0.4	0.6	1.2	2.7	
⑥ BANCSセンダIn→BANCSセンダOut	0.1	0.1	0.8	4.6	4.3	6.7	27.2	
⑦ BANCSセンダOut→シミュ→BANCSリスナIn	719.8	740.7	829.9	1253.7	1133.4	1056.6	1107.8	
⑧ BANCSリスナIn→Out	0.0	0.0	0.0	0.0	0.0	0.0	0.1	
⑨ BANCSリスナOut→BANCS AP1 In	0.2	0.2	0.2	0.4	0.6	1.2	4.0	
⑩ BANCS AP1 In→Out	48.0	51.2	58.5	65.2	71.2	89.9	**148.4**	*
⑪ BANCS AP1 Out→MQ I/F In	0.1	0.2	0.3	0.4	0.5	1.1	2.9	
⑫ MQ I/F In→Out	0.1	0.1	0.1	0.2	0.3	0.4	0.6	
⑬ MQ I/F Out→シミュレータ	4.1	4.2	4.2	4.1	4.9	5.1	5.5	

この間のプロセス毎の CPU 使用率に着目すると， BANCS AP は増加しておらず，SQL Server が 9.4% 増加していることがわかる（図 8）．このことから，BANCS AP 内の処理時間の増加は，SQL Server 内での処理時間が増加しているためと思われる．

以上から，回線数を増加しても，SQL Server 内の処理時間が増加していくと予想され，処理量の大幅な増加は期待できない．

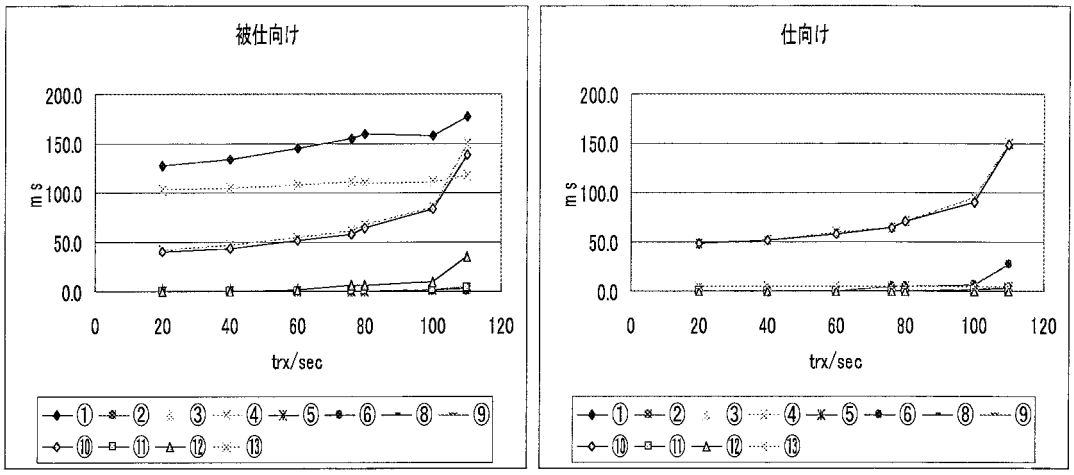

図 7　各プロセスの処理時間

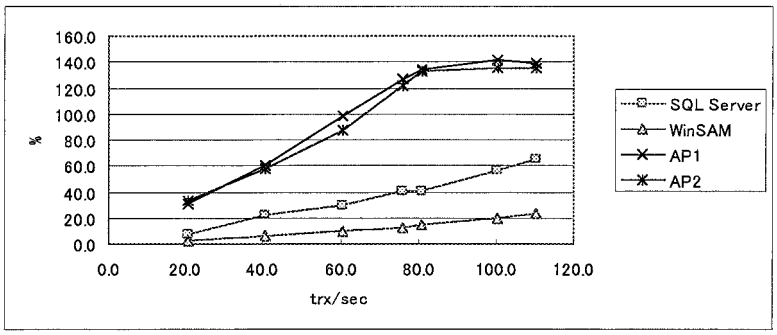

図 8　プロセス毎の CPU 使用率

5.3.3　CPU Affinity の設定による違い

前述したが，メモリ間データ転送を頻繁に行なうプロセス同士は Third　Level Cache を共用するように CPU　Affinity を設定することにより性能が良くなる．どのような効果があるのかを検証した結果が表 12 である．CPU　Affinity は，表 7 のように設定した．

この結果から，適切に CPU　Affinity を設定することで，スループットを向上させながら，CPU 使用率を 20～25% 低下させることが可能であることが検証できた．

なお，この結果は，擬似 AP,UST なしで測定した結果なので，表 8 の数値とは異なる．

6.　問題点と対処

本章では，性能検証作業の間に遭遇した主な問題点とその対処方法について記述する．

1）　ES 7000 での高い CPU 使用率

当初，ES 7000 での CPU 使用率が高く，性能要件を満足することができなかった．これについては，5.3.3 項で述べたように，CPU Affinity の設定で解決した．

表 12　CPU Affinity の設定有無による性能比較

	通常		被災		Failover		被災＆Failover	
Affinity	なし	あり	なし	あり	なし	あり	なし	あり
要件定義								
業務件数／秒	16.5	16.5	25	25	33	33	50	50
trx/sec	33	33	50	50	66	66	100	100
計測結果								
業務件数／秒	15.8	17.1	25.3	25.6	32.9	35.0	50.0	51.5
被仕向け	10.2	11.0	15.2	15.2	19.5	22.0	30.9	30.9
仕向け	5.6	6.1	10.1	10.4	13.4	13.0	19.1	20.6
SQL Serverのtrx/sec	31.6	34.5	50.9	51.6	66.2	70.5	101.7	103.4
ABCG101	31.6	34.5	50.9	51.6	33.2	35.6	50.9	52.1
ABCG102					33.0	34.9	50.8	51.3
滞留時間(ms)								
被仕向け	184	134	154	145	201	210	901	910
仕向け	207	148	184	170	233	265	1039	960
CPU使用率(%)								
BANCSサーバ	22.2	16.8	35.5	26.9	51.5	40.5	90.5	71.7
User	12.1	9.2	20.0	14.4	30.1	22.6	50.6	36.4
SQL Server	21.5	16.4	37.8	24.1	30.4	19.4	56.2	36.1
WinSAM	9.2	3.0	16.1	9.1	21.6	13.1	43.8	25.9
BANCS AP1	46.6	37.5	59.3	44.5	61.1	47.6	75.2	53.0
BANCS AP2	44.3	36.8	53.3	42.0	54.8	40.8	70.5	48.3
BANCSリスナ	0.9	0.9	0.6	0.6	0.9	0.8	1.2	1.0
BANCSセンダ	0.7	0.6	0.9	0.7	0.5	0.8	1.9	1.3
MQFEP I/F	2.0	1.4	1.8	1.0	1.4	1.3	1.9	1.5
Processor 0	20.4	21.4	34.6	27.0	50.4	43.4	89.4	76.7
Processor 1	17.4	13.7	32.7	25.5	49.8	40.7	90.1	74.0
Processor 2	24.7	19.2	34.0	28.3	52.0	43.9	90.6	75.1
Processor 3	21.1	17.3	35.3	28.7	49.1	43.3	91.1	74.2
Processor 4	20.1	11.3	32.0	23.5	48.9	36.7	90.4	66.2
Processor 5	19.5	11.8	33.3	23.7	51.5	35.0	90.8	67.0
Processor 6	27.3	16.2	40.2	27.4	53.6	41.4	90.8	70.9
Processor 7	27.2	23.5	41.9	31.4	56.5	39.8	91.7	69.8
平均	22.2	16.8	35.5	26.9	51.5	40.5	90.6	71.7
Context Switch/sec	7209	6743	9250	8891	10296	10621	13763	16195

2）　WinSAM での低スループット

プロトタイプ 1 で検証した時点の WinSAM　V 4 M 2 版は，開発元では Windows 2000 上での動作を保証していなかった．プロトタイプ 2 での検証の時期に，Windows 2000 正式サポート版である V 5 M 0 がリリースされたが，V 4 M 2 に比べ処理能力が極端に悪く（V 4 M 2 の半分程度），とても使えるものではなかった．計測結果を開発元にレポートして協議した結果，V 5 M 0 を捨てて V 4 M 2 をベースにした V 5 M 1 として作り直すことになった．改善点は以下のとおり．

・V 5 M 0 では VC＋＋の MFC を使用したオブジェクトベースのアーキテクチャを採用したが，V 4 M 2 で採用していた関数ベースのアーキテクチャに戻した．

・V 4 M 2 ではマルチプロセス構造であったが，シングルプロセスのマルチスレッド構造にした．

・Windows の TCP/IP 関数が，短い電文は溜まってからまとめて送信する Nagle アルゴリズムを採用していたため，電文長が 100 Byte の本システムの場合，効率が悪かった．Nagle アルゴリズムを無効にするオプションを設けた．

・内部で使用しているパイプ数が 10 個固定であったのを増やせるようにした．

3）　UST 使用時の低スループット

UST 経由で性能検証したところ，最初は全くスループットがでなかった．原

因を追求した結果，UST 内の同じ通信ボードで複数回線使用しても，1 回線分の能力しか出ていないことがわかった．

UST 内の通信ボードは，1 枚のボードで MAX 4 回線まで通信することが可能である．1 回線だけの最大スループットは，表 10 に示したように 9.15 trx/sec であった．2 回線での限界値を測定したところ，違うボード上の回線を使用した場合は 18.26 trx/sec とほぼ 2 倍の処理をこなせたが，同一ボード上の回線の場合は 9.96 trx/sec で，ほとんど増加しなかった．

この問題は，WinSAM の定義ファイルを変更することで改善することができた．回線定義で回線ごとに NopFCH というオプションを設定することで，同一ボード上の回線を使用した場合もスループットを増やすことができるようになった．

4） SQL Server のネットワークプロトコル優先順位

SQL Server クライアントネットワーク設定でネットワークプロトコルの優先順位を①「名前付きパイプ」，②「TCP/IP」とすると CPU 使用率が高くなりスループットも 65 trx/sec までしか上がらなかった．優先順位を②から①へと逆転させる CPU 使用率が半分になりスループットも上がった．

5） SQL Server のテーブルファイル拡張時のオーバヘッド

2000 万件のデータを生成した際に，平均 1300 trx/sec で Insert されていたが，時々 150 trx/sec に落ちてしまう現象が発生した．これは，テーブルを生成する際の初期サイズが小さい値であったため，Insert 時に領域が不足し，自動拡張が行われていたためであった．

データベースのテーブルは，あらかじめ最大件数分の領域を確保しておき，自動拡張が発生しないようにしておく必要がある．

7. おわりに

最新の技術を使用した ES 7000 と W 2 KDCS を使用したシステム構築は，換言すれば，実績の無いプラットフォームが抱えるリスクへのチャレンジそのものと言えた．

リスク回避のために，できるだけ早期にプラットフォームの品質（性能，機能，安定性）の大枠を掴み，発生した問題に適切な処置を行なうことができるよう，プロトタイピングやシミュレータ開発を絡ませ，段階的に検証を積み重ねた．結果，ともすれば開発の終盤に発生しがちな性能（効率）問題を統合テスト時点でリスク項目から排除できることが判り，それ以外の領域にパワーをかけることが可能となって，開発期間内のシステム完成に大きな弾みをつけた．

本稿が，ES 7000 と W 2 KDCS 上でのシステム構築の際，および，プロジェクトで POC を実施しようとする際の一助となれば幸いである．

最後に，本システムの構築に際し，ご指導，ご協力をいただいた M 銀行の皆様方，ならびに社内関連部署の方々に謝意を表します．

執筆者紹介　山　岸　重　雄（Shigeo Yamagishi）
1983 年慶応義塾大学経済学部卒業．同年日本ユニシス（株）入社．主にクライアント・サーバ系のシステム開発に従事．現在インテグレーションサービス部 .NET ビジネスディベロプメントに所属．